ISSN 0278-6419, Moscow University Computational Mathematics and Cybernetics, 2021, Vol. 45, No. 4, pp. 135–147. © Allerton Press, Inc., 2021.
Russian Text © The Author(s), 2021, published in Vestnik Moskovskogo Universiteta, Seriya 15: Vychislitel'naya Matematika i Kibernetika, 2021, No. 4, pp. 3–16.

User Behavior Authentication Based on Computer Mouse Dynamics

A. V. Berezniker[1*], M. A. Kazachuk[1], I. V. Mashechkin[1***], M. I. Petrovskiy[1****], and I. S. Popov[1*****]**

[1] *Faculty of Computational Mathematics and Cybernetics, Moscow State University, Moscow, 119991 Russia*

Received August 9, 2021; in final form, August 31, 2021; accepted August 31, 2021

Abstract—The aim of this work was to investigate existing algorithms for dynamic user authentication and develop our own, based on an analysis of computer mouse handling characterized by high-quality performance and a supporting dynamic mode. Existing ways of constructing and preliminarily processing the feature space are considered, along with means of dynamic authentication based on the use of classical techniques of machine learning and neural networks. Modifications with better efficiency are proposed for the ones most promising. A cross-platform application for dynamic user authentication based on an analysis of computer mouse handling is designed and implemented using the proposed technique, which shows the best performance (ROC AUC = 0.82). This system and its individual modules can serve as a basis for building advanced information security systems. Experimental studies are conducted that confirm the validity of the obtained results and the correctness of the system's operation.

Keywords: dynamic authentication, biometrics, single-class classification, gradient boosting, dynamic local outlier removal, single-class support vector technique, neural networks, dynamic trust level change.

DOI: 10.3103/S027864192104004X

1. INTRODUCTION

Information systems have become an integral part of different spheres of human activity. An enormous amount of confidential information is transferred, stored, and processed in information systems, making it necessary to ensure that they are reliably protected.

People use such access control mechanisms as passwords, keycards, or biometrics to protect against unauthorized access by another person. This means the user must provide proof of identity when starting or unlocking the system. Personal computer access control is usually done as a one-time confirmation of identity during initial authorization. Authorization is the procedure of granting a subject certain rights of access to system resources after they have successfully passed the authentication procedure. It is assumed that only the authorized (legitimate) user will be in the system throughout a session. In many cases, however, people leave their computers unattended while temporarily away from their workplaces, and anyone can access the same data sources with the same rights as a legitimate user.

Information security in information systems is ensured by creating a comprehensive security system, one of whose main components are ways of protecting against unauthorized access [1, 2]. The basis of software and technical means of protection against unauthorized access are procedures for identifying and authenticating users. The identifier is in this case a unique feature of the object, which allows us to distinguish it from other objects. By authentication procedure, we mean the process of checking whether

[*] e-mail: berezniker@mail.ru.
[**] e-mail: mkazachuk@cs.msu.ru.
[***] e-mail: mash@cs.msu.su.
[****] e-mail: michael@cs.msu.su.
[*****] e-mail: ivan@jaffar.cs.msu.su.

the identifier presented by an access subject belongs to them. Identifiers used in existing systems: secret knowledge (e.g., password or key) and physical objects belonging to users (e.g., flash drive or keycard) are easily stolen or compromised, so they are replaced with authentication systems based on biometric data of users (i.e., unique biological and physiological characteristics that allow us to establish a person's identity). In contrast to the identifiers listed above, biometric samples cannot be forgotten or lost and are much harder to compromise.

Current means of biometric authentication can be based on physiological characteristics of a person that they bear throughout their life, or on behavioral characteristics of a person. The latter are characteristics of individual behavior and are distinguished by relative stability and consistency in how they are displayed. The main disadvantage of user verification based on physiological biometrics is that they require such hardware devices as fingerprint sensors and retinal scanners, which are expensive and not always available. Although fingerprint verification is becoming widespread in laptops and smartphones, it is still not universally popular and cannot be used in web applications. In addition, fingerprints can be copied. Means based on behavioral biometrics in turn do not require special equipment, since they use such common input devices to collect biometric data as mouses and keyboards.

Depending on the principle of an identity verification procedure, there are static (occasional verification of the user identity) and dynamic (user identity is checked continuously throughout a computer session) means of authentication. Dynamic user authentication is obviously preferable, since it excludes scenarios in which an intruder gains access to the information system after a legitimate user has been authenticated. However, this approach consumes more computer resources due to its continuous operation.

We therefore see the problems of the lack of control over changes in user and compromised identity, which we propose solving by means of biometric user behavioral characteristics for dynamic authentication. Dynamic user authentication based on analyses of computer mouse handling is an important line of research in the field of computer security. An advantage of this approach is its ease of use: only an input device (a computer mouse) and special analysis software are required. However, existing solutions in this area have a number of disadvantages, particularly the supervised collection of data, the use of binary classifiers, and retraining on the initial datasets. We therefore focus on a context-independent dynamic authentication system that responds to each individual action performed by a user.

Section 2 of this work gives an overview of existing solutions on this subject. Section 3 focuses on the proposed approaches to building the feature space. Section 4 describes the proposed ways of constructing the user model. Section 5 is devoted to an experimental study of the proposed means. Section 6 describes the architecture of the developed cross-platform dynamic user authentication application. Section 7 reaches conclusions on the proposed algorithms.

2. OVERVIEW OF EXISTING SOLUTIONS

Different approaches to collecting experimental data in the literature differ in the number of users who participated in the study, the means of acquisition, and the size of the collected data. However, the greatest difference is the environment in which the data are collected. For example, it can be supervised (the user performs such assigned tasks as clicking on objects that appear on the screen or working only with text data) or unsupervised (the user works in a familiar environment and performs their daily tasks). Collecting experimental data in a supervised environment with a specific task for the user, perhaps even on a specific computer, has serious drawbacks. The user will in this case be more focused on the task, and their behavior will not correspond to their normal state. The results from experiments in a supervised environment thus cannot be generalized to ones in the real world. However, most existing works on the subject consider only supervised data collection.

Features of mouse trajectory most commonly used in scientific works are kinematic characteristics (displacement, trajectory length, velocity, and acceleration), direction of motion, and curvature of a motion's trajectory. These features are calculated from certain collected characteristics: type of action (key press/release and movement of mouse), mouse button state, cursor coordinates, and timestamp. Normalization is often used as a way of preliminarily processing the feature space [3]:

$$\hat{x} = \frac{x - \mathbb{E}x}{\sqrt{\mathbb{D}x}}, \tag{1}$$

Table 1. Performance of existing solutions

Paper	N	Dataset	Model	Unsupervised learning	ROC AUC
[3]	10	BALABIT	Linear SVC	×	0.81
[4]	28	Unknown	SVM	×	0.85
[5, 6]	14	BALABIT+TWOS	2D-CNN	×	0.88
[7]	25	Unknown	Autoencoder	✓	**0.91**
[8]	50	Unknown	One-Class SVM	✓	**0.93**

where $\mathbb{E}x$ is the expectation of an observation and $\mathbb{D}x$ is its variance.

The performance of existing solutions that consider the work of users in an unsupervised environment is summarized in Table 1, where N is the number of users who participated in the experiment. As can be seen from Table 1, most works consider binary classification (supervised learning), but in a real situation there can often be no examples of data belonging to an illegitimate class. We must therefore use algorithms of single-class classification. The best recognition in existing works is shown by one-class support vector machines and autoencoder neural networks.

3. CONSTRUCTING THE FEATURE SPACE

During our preliminary analysis of data characterizing the dynamics of users' work with a mouse, we found completely duplicate entries, duplicate timestamps, multiple duplicate mouse positions (looping), and abnormally high coordinate values.

All of these features were eliminated in preliminary processing:

- Duplicate entries were removed because they could be the result of failures in the data acquisition system on user devices and result in incorrect calculations of a feature's characteristics.
- Duplicate timestamps were replaced with averages of neighboring timestamps to bypass division by 0 in time-related features:

$$\text{timestamp}_i = \frac{\text{timestamp}_{i-1} + \text{timestamp}_{i+1}}{2};$$

- Multiple duplicate mouse positions (fixed cursor coordinates (x, y) for changing time t) were omitted because such entries can indicate a pause in the user's work and must be considered by the model classifier.
- The abnormally high coordinate values were replaced with the value of the upper bound of the interquantile analysis. The interquantile range (IQR) was the difference between the 75th and 25th quantile. Observations beyond the upper and lower boundaries were empirical outliers.

The sequences of events of user's work with the mouse, tuples of the form <activity type, mouse button state, cursor coordinates, timestamp> were then divided into time windows, and for each of which a feature vector was constructed containing statistical information about the events in it. It was proposed a concatenation of feature sets from [8, 9] be used to construct feature spaces, which describe aspects of a mouse's trajectory. The resulting feature spaces had dimensions of 78 features per vector:

- Velocity, acceleration, displacement, and length of a trajectory, along with their discretized, minimum, and maximum characteristics, mean values, and standard deviations.
- Direction of movement (a total of eight directions were considered).

- Average curvature of movement trajectories: $\frac{1}{n}\sum_{i=0}^{n}\frac{\angle P(x_i,y_i)P(0,0)P(x_i,0)}{\sqrt{x_i^2+y_i^2}}$, where $P=(x_i,y_i)$ are cursor position coordinates.
- Trajectory of the center of mass: $TCM=\frac{1}{S_{n-1}}\sum_{i=1}^{n-1}t_{i+1}\sqrt{(x_{i+1}-x_i)^2+(y_{i+1}-y_i)^2}$, where t_i is the timestamp and S_{n-1} is the length of the path.
- Coefficient of scattering: $SC=\frac{1}{S_{n-1}}\sum_{i=1}^{n-1}t_{i+1}^2\sqrt{(x_{i+1}-x_i)^2+(y_{i+1}-y_i)^2}-TCM^2$.

For the processing of feature space, we proposed normalization and selection of the most important features using gradient boosting [10]. Processing features with One-Hot Encoding and discretization by quantiles [11] was not considered in this work, since use of these procedures did not improve the quality of recognition at the preliminary stage of our experimental studies.

Boosting is a way of gradually constructing a set of machine learning algorithms in which each successive algorithm seeks to compensate for the flaws in the composition of all previous algorithms. Let us present classification function f as a composition of T functions such that each of the subsequent functions minimizes the residuals from the one before it. We modify the model using a descending gradient because modifying f based on outliers is similar to modifying f based on a negative gradient.

The gradient boosting algorithm is then built by:

- constructing the initial model $f(x)$;
- calculating the antigradient in the loop from $t=1\ldots T$, according to the depth of the composition;
- building regression h_t according to the antigradient;
- modifying the model: $f(x)\leftarrow f(x)+\rho_t\cdot h_t(x)$, where ρ_t is the gradient step.

The advantage of using ensembles of such decision trees as gradient-based boosting is that they can estimate the importance of features from the trained model. Importance usually provides a score that indicates how useful each feature was in building decision trees in the model. The more a feature is used to make key decisions, the higher its relative importance. Importance is calculated for each decision tree, and feature values are then averaged over all decision trees in the model. This approach requires the availability of tagged data, so it was used only to reduce dimensionality and filter the feature space during the experimentation phase.

4. CONSTRUCTION A USER MODEL

The problem considered in this article belongs to the class of finding anomalies in data (unsupervised learning). An object or event in a sample is anomalous if its features do not correspond to dependences typical of other objects or events in this sample. To solve this problem, we considered three single-class classification algorithms: Support Vector Clustering (SVC), Isolation Forest, and Elliptic Envelope.

Since the training sample can itself contain anomalies, we propose first using the Local Outlier Factor [12] technique to rid each sample from anomalous observations. It detects outliers (anomalous observations) based on the local density of points that are outliers with respect to their local neighborhood, but not to the global data distribution. The higher the LOF value for an observation, the more anomalous it is. A point is considered anomalous if its local density differs appreciably from the density of neighboring points. The LOF at point P is defined as

$$\mathrm{LOF(P)}=\sum\nolimits_1(\text{distance to neighbors})\cdot\sum\nolimits_2(\text{local density of neighbors}). \quad (2)$$

An LOF at point P can therefore take

- A high value if point P is far from its neighbors and its neighbors have high local density (i.e., they are close to one another).
- A medium value if point P is far from its neighbors and its neighbors have low local density.
- A low value if P is close to its neighbors and its neighbors have low local density.

The metaparameters of this algorithm are $n\_neighbors$, the number of neighbors, and $contamination$, the proportion of anomalies in the sample.

In SVC [8], the objects from the initial set are implicitly mapped using a potential function into a high-dimensional feature space, where a hypersphere of minimal radius is then sought that contains "the main part" of images of objects from the initial set. Anomalies are objects whose images lie outside the identified hypersphere, solving the problem of optimization:

$$\min_{\xi\in\mathbb{R}^N, R\in\mathbb{R}, a\in H}\left[R^2+\frac{1}{\nu N}\sum_{i=1}^{N}\xi_i\right],$$

$$||\varphi(x_i)-a||_H^2\leqslant R^2+\xi_i\quad\forall i\in[1,N], \tag{3}$$

where $\varphi(x_i)$ is an image of the initial object x_i in the feature space of high dimension H; R is the radius of the constructed hypersphere; a is the center of the hypersphere; N is the number of objects in the training sample X; $0<\nu\leqslant 1$ is the predefined percentage of anomalies, and ξ_i denotes additional variables.

The adjustable parameters of this algorithm are $kernel$, the type of the potential function, and ν, the ratio of the expected number of anomalies to the total number of objects in the considered sample.

Note that One-Class SVM is another variation of the one-class support vector machine. It builds a hyperplane separating normal and anomalous data, instead of building a hypersphere in the high-dimensional feature space. Note that the results from SVC and One-Class SVM are identical when using radial-basis potential function (rbf).

Like any other tree ensemble, an isolation forest [13] is built on decision trees. The learning algorithm builds an ensemble of isolation trees based on a recursive and randomized structured partitioning procedure in which an object is first selected randomly, and a random value between the minimum and maximum ones of the selected object is chosen. There are few anomalies in the data, and they are often farther away from the normal observations in the feature space. They must therefore be identified closer to the root of the tree when using this random partitioning. Fewer divisions are needed for anomalous observations. To decide whether an observation is anomalous, this algorithm uses the function

$$S(x,n)=2^{-\frac{\mathbb{E}(h(x))}{c(n)}}, \tag{4}$$

where $\mathbb{E}(h(x))$ is the average length of an observation's path x; $c(n)$ is the average length of the path of an unsuccessful search in the binary search tree; and n is the number of external nodes.

Each observation is given an anomaly index on whose basis a decision is made. A score close to 1 indicates the observation is anomalous. A score close to 0 indicates normal behavior of the observation. This algorithm has the metaparameters $n\_estimators$, the number of trees; $max\_samples$, the size of the sample for building one tree; and $contamination$, the proportion of anomalies in the sample. It is robust to the curse of dimensionality and is highly efficient.

Ellipsoidal approximation of the data [14] is a covariance estimate that assumes the data have a Gaussian distribution. The bounding box is elliptical in shape. The FAST-Minimum Covariance Determinate algorithm is used to estimate the size and shape of the ellipse. This algorithm selects nonoverlapping subsamples of data and computes mean μ and covariance matrix C in the feature space for each subsample. Mahalanobis distance dMH is calculated for each multivariate data vector (feature) x in each subsample using the formula

$$dMH=\sqrt{(x-\mu)^T C(x-\mu)}. \tag{5}$$

The data are then ordered according to the ascending values of dMH. This procedure is repeated until the determinant of the covariance matrix converges. The covariance matrix with the lowest determinant of all subsamples forms an ellipse that covers part of the initial data. The data within the surface of the

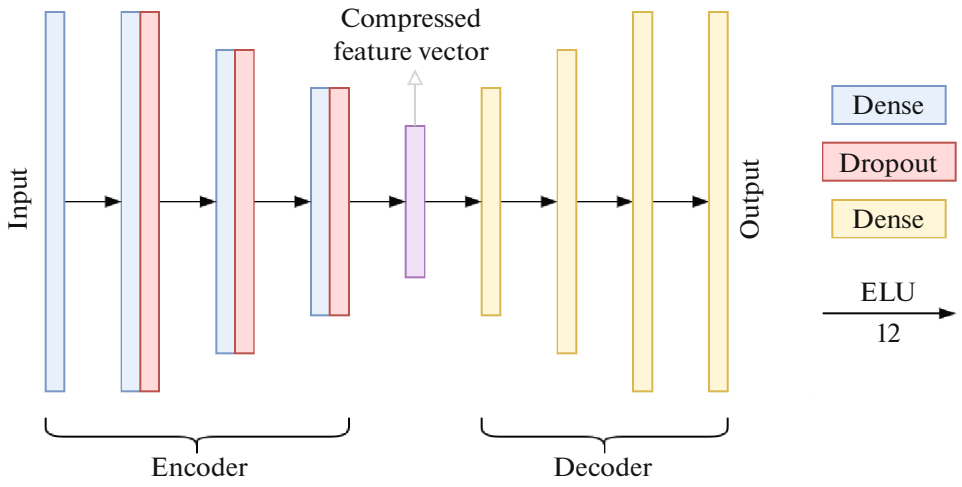

Fig. 1. Proposed architecture of the fully-connected autoencoder.

ellipse are normal. Outside the ellipse, they are anomalous. The metaparameter of this algorithm is *contamination* (i.e., the proportion of outliers in the sample). However, this approach is successful only with normally distributed unimodal data.

Autoencoder [7] is a neural network that reconstructs the input signal on the output. Mathematically, each neuron of the neural network is parameterized function $f_{\omega_1,\ldots,\omega_m}(x_1,\ldots,x_n,b)$ that returns a single value, that of the output signal. Parameters $\omega_0,\ldots,\omega_m$ of function $f_{\omega_1,\ldots,\omega_m}(x_1,\ldots,x_n,b)$ are the synaptic weights of the neuron. A linear combination of neuron inputs $x_1,\ldots,x_n$ and offset b, which is also a parameter of function $f_{\omega_1,\ldots,\omega_m}(x_1,\ldots,x_n,b)$, are most often used for the aggregation of inputs. The aggregate value of neuron inputs is fed to the input of the neuron activation function, which calculates the value of the output signal directly. Interconnected neurons form a neural network. If the neural network consists of a great many neurons, we also introduce the concept of neural network layers and divide them into three types: input, hidden, and output. A vector of input values (also called a vector of features) is fed to the input layer of the neural network, and a vector of output values (also called the prediction of the neural network) appears on the output layer.

The autoencoder consists of two parts: an encoder, which encodes data into its internal representation, and a decoder, which reconstructs the initial vector. Autoencoders are usually limited in the dimensionality of hidden layers (they are smaller than that of an input signal). We have proposed several neural network architectures for solving the problem. In this work, we used L_2-regularization and dropout-layers to prevent overtraining.

L_2-regularization is performed by imposing penalties on the weights with the highest values, minimizing their L_2-norm using the λ regularization factor. For each weight ω, we add the summand to target function $\mathcal{L}$:

$$\frac{\lambda}{2}\omega^T\omega = \frac{\lambda}{2}||\omega||_2 = \frac{\lambda}{2}\sum_{i=1}^{n}\omega_i^2. \tag{6}$$

The idea of dropout is that instead of training a single neural network, we train several and make an ensemble of them, while averaging of the results. Such networks are obtained by eliminating neurons with probability p. Neuron exclusion means this returns 0 for any input.

Figure 1 shows the proposed architecture of an autoencoder network with seven hidden layers. The ELU activation function is used before each fully-connected layer (7):

$$\mathrm{ELU}(x) = \begin{cases} x, & x > 0, \\ e^x - 1, & x \le 0. \end{cases} \tag{7}$$

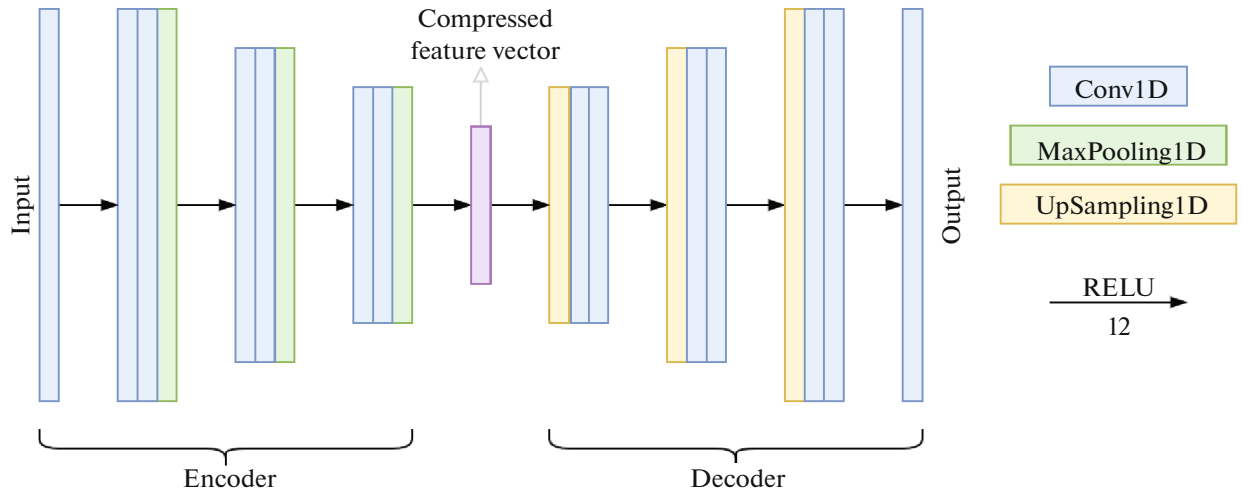

Fig. 2. Proposed architecture of the convolutional autoencoder.

We used the Huber function as the loss function because it is robust to outliers:

$$L_\delta(x) = \begin{cases} \frac{1}{2}x^2, & |x| \le \delta, \\ \delta(|x| - \frac{1}{2}\delta), & |x| > \delta. \end{cases} \tag{8}$$

To decide whether a user is legitimate, we determine the value of the metric MSE (9) between the feature vector at input (y^{true}) and output (y^{pred}) of the autoencoder. The better the network reconstructs the vector, the smaller the error, and we therefore assume that the model's confidence in the legitimacy of the current user is greater.

$$\mathrm{MSE}(y^{true}, y^{pred}) = \sqrt{\sum_{i=1}^{n}(y_i^{true} - y_i^{pred})^2}. \tag{9}$$

We also propose a neural network with a convolutional autoencoder architecture (CNN-Autoencoder) shown in Fig. 2. The main feature of the fully-convolutional neural networks is a relatively small number of parameters due to the use of convolutional layers. This allows us to design and train more complex architectures, in order to improve the quality of the solution to the problem. In this architecture, we also used L_2-regularization and dropout layers to prevent overtraining. The RELU function was selected as the activation function (10):

$$\mathrm{RELU}(x) = \max(0, x). \tag{10}$$

We propose considering the trust model ([8]) a dynamic component of the authentication system. The basic idea of this technology is that the trust (or confidence) of the system in the authenticity of the current user depends on their deviations from the normal state. The system's trust in the authenticity of the current user is increased (a reward) if the current action is performed according to a pattern stored in the profile of the legitimate user. The system's trust in that user is reduced (a penalty) if there is a noticeable difference between the behavior of the authentic and current users. The number of changes in the level of trust can be fixed or variable, and the value of the penalty or reward is set by the delta-function.

No one always behaves in just one way. For a legitimate user, this means they will also sometimes deviate from their normal (template) behavior, resulting in a lower level of trust. However, most actions of a legitimate user will be close to their normal behavior; i.e., they will increase the level of trust. This will generally create a high level of trust in the system. However, the opposite is true for an attacker. Only in some cases will they be able to behave like the legitimate user and increase their level of trust, but most of their actions will reduce it because of large deviations from the behavior of the legitimate user. This eventually leads to an overall drop in trust over time and lockout.

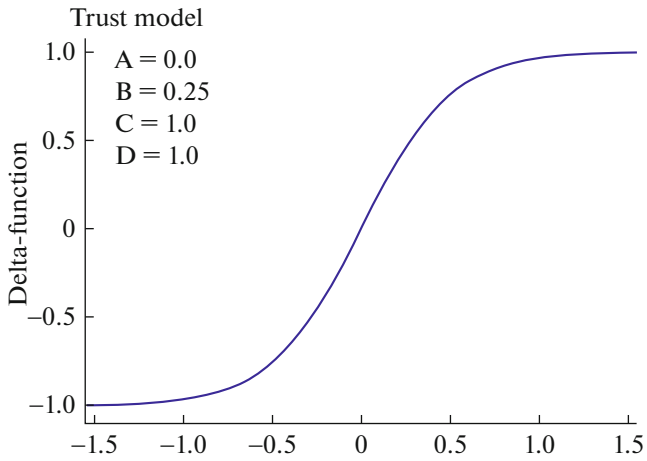

Fig. 3. Delta function of the dynamic change in the confidence level.

In the trust model algorithm,

———————————————————————————————

Algorithm Trust Model

———————————————————————————————

score_i is the response of the base classifier for the ith action;

$Trust_i$ is the confidence of the algorithm for the ith action;

$Trust_{lockout}$ is the lockout level;

A is the threshold for penalty or reward;

B is the sigmoid width, $B > 0$;

C is the maximum reward, $C > 0$;

D is the maximum penalty, $D > 0$.

$Trust_i > Trust_{lockout}$

$\text{score}_i = \text{model.score}(\text{data}_i)$ is the calculation of the response $score_i$ of the classifier $model$ on a new

portion of data $data_i$;

$$\Delta_T(\text{score}_i) = \min\left\{-D + \left(\frac{D \times \left(1 + \frac{1}{C}\right)}{\frac{1}{C} + \exp\left(-\frac{\text{score}_i - A}{B}\right)}\right), C\right\};$$

$$Trust_i = \min\{\max\{Trust_{i-1} + \Delta_T(\text{score}_i), 0\}, 100\};$$

Loop end

———————————————————————————————

Note that delta function $\Delta_T(\text{score}_i)$ is a sigmoid with parameters A, B, C, D. The values selected for a smooth change in user confidence were $A = 0.0$, $B = 0.25$, $C = 1.0$, $D = 1.0$. The graph of this function is shown in Fig. 3.

Table 2. Data sets for experimental studies

Dataset	Number of users	Number of events per user	Recording time per user
BALABIT	10	700 000	43 h
DATAIIT	20	1 150 000	20 h
TWOS	4	400 000	10 h
DFL	21	7 400 000	100 h
CHAOSEN	50	200 000	4 h

Table 3. Metaparameters of machine learning algorithms

Machine learning method	Algorithm metaparameters
SVC	kernel = "rbf" $\nu = 0.1$
Isolation Forest	n_estimators = 10 max_samples = 0.7 contamination = 0.1
Local Outlier Factor	n_neighbors = 10 contamination = 0.1
Elliptic Envelope	contamination = 0.1

5. EXPERIMENTAL

We found four open datasets collected in an unsupervised environment: BALABIT, used in [3, 5, 6], TWOS [5, 6], DFL [15], and CHAOSHEN [16]. The authors' datasets were used in the remaining works, along with ones not generally available. We also collected our own independent dataset, DATAIIT. The characteristics of these datasets are presented in Table 2.

Each data set was divided into training and test parts, 75% and 25% of the initial data, respectively. Each part contained records of authorized sessions of all users from set $\mathfrak{U} = \{U_1, \ldots, U_j, \ldots, U_q\}$. The users' work was divided into sessions of different duration $U_j = \{S_1, \ldots, S_i, \ldots, S_m\}$, where each session $S_i \in U_j$ $(i = \overline{1,m},\ j = \overline{1,q})$ contained entries in the form $(\text{time}, x_{pos}, y_{pos})$, where time is the timestamp and (x_{pos}, y_{pos}) are the coordinates of cursor positions. We then divided each session into segments with the time threshold constraint from above and the minimum number of actions in this time interval from below. A single feature vector was constructed from the resulting segment. Analysis of the datasets showed it was best to partition them into segments of 3–5 s. Each of these contained a sufficient number of user actions for uniqueness of the features, and also allows us to frequently check the user. The number of feature vectors in the training sample was also quite high (about 3500 vectors for each user). All the identified datasets were combined as part of our experiments; i.e., a one-against-all test was performed for each of the 105 users of the resulting set.

Our experimental study found that feature space normalization contributed an average of 0.05 to the ROC AUC metric for speeding up and improving the quality of the approach. Additional improvement in recognition (0.02 ROC AUC) was made with feature selection based on gradient boosting. The dynamic detection of unsupervised local outliers based on the Local Outlier Factor technology before model training resulted in an average improvement of another 0.02. Metaparameters for the machine learning algorithms were selected via greedy search. We thus selected the values (see Table 3).

Table 4. Experimental results

	Normalization	Gradient boosting	Outlier removal	Mean ROC AUC	Median ROC AUC ± IQR
SVC	−	−	−	0.732	0.750 ± 0.032
	−	−	+	0.752	0.770 ± 0.031
	−	+	−	0.752	0.770 ± 0.029
	−	+	+	0.772	0.790 ± 0.045
	+	−	−	0.782	0.800 ± 0.043
	+	−	+	0.802	0.820 ± 0.021
	+	+	−	0.802	0.820 ± 0.025
	+	+	+	**0.822**	**0.840 ± 0.038**
CNN-Autoencoder	−	−	−	0.727	0.747 ± 0.031
	−	−	+	0.747	0.767 ± 0.027
	−	+	−	0.747	0.767 ± 0.021
	−	+	+	0.767	0.787 ± 0.041
	+	−	−	0.777	0.797 ± 0.026
	+	−	+	0.797	0.817 ± 0.045
	+	+	−	0.797	0.817 ± 0.042
	+	+	+	**0.817**	**0.837 ± 0.034**
Autoencoder	+	+	+	0.732	0.749 ± 0.061
Isolation Forest	+	+	+	0.655	0.673 ± 0.072
Elliptic Envelope	+	+	+	0.542	0.543 ± 0.101

The final results from our experiments are presented in Table 4. The mean and median values of the area under the ROC curve (ROC AUC) and the interquartile range (IQR) were used as metrics. An advantage of ROC AUC is that its value does not depend on the selected threshold, which determines how close to the threshold the output of the classification algorithm must be for it to coincide. The best result was obtained using a combination of procedures:

- Normalization of the feature space.
- Gradient boosting to identify significant features.
- Dynamic removal of local outliers in the training sample based on the Local Outlier Factor technology.
- One-class SVC support vector machine as a classifier.

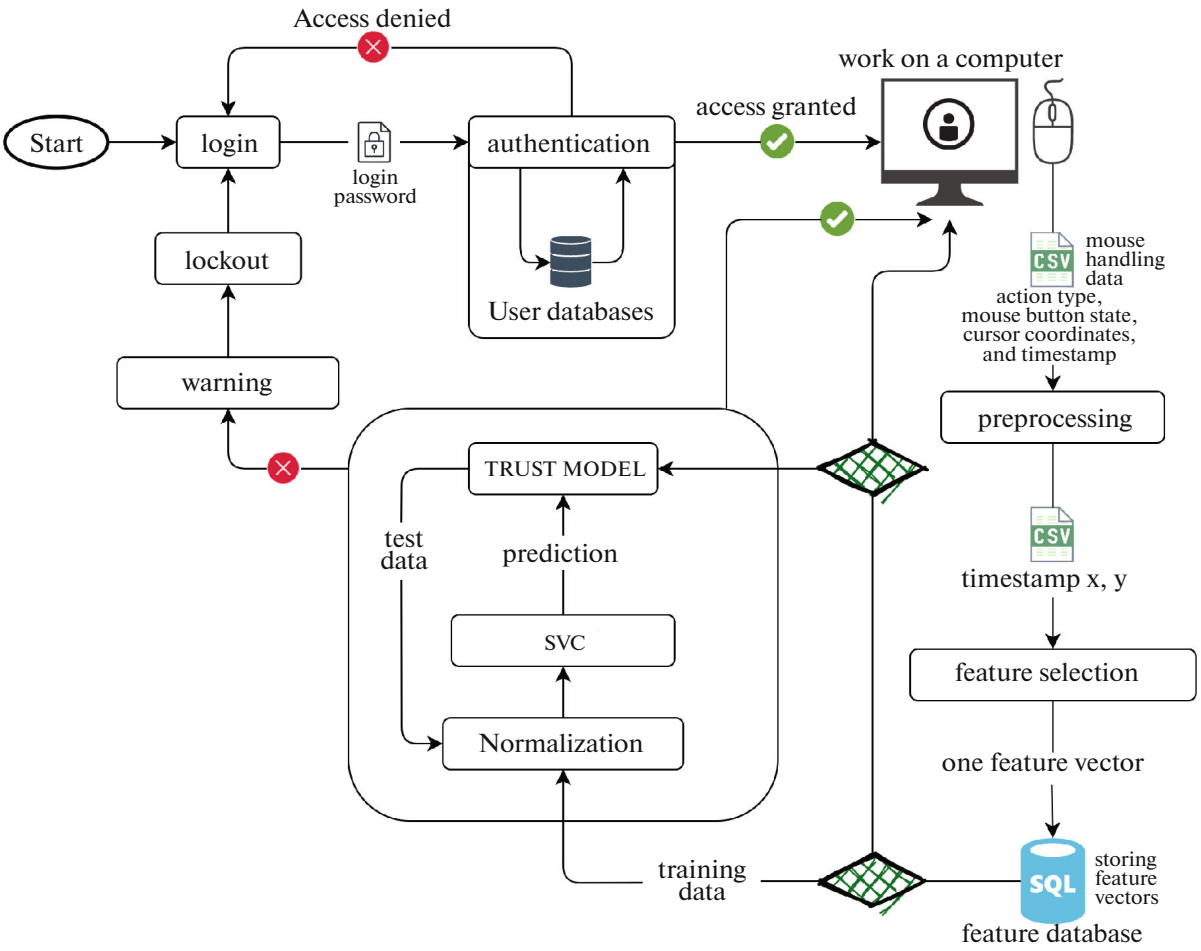

Fig. 4. Cross-platform application architecture.

6. CROSS-PLATFORM APPLICATION OF DYNAMIC USER AUTHENTICATION

The architecture presented in Figure 4 was designed on the basis of our experimental results. We also created a cross-platform application for dynamic user authentication based on an analysis of computer mouse handling. Even though the proposed convolutional autoencoder architecture was almost as good as the SVC algorithm in terms of quality of recognition, a single-class support vector machine was selected as the basic classifier because less time and computing resources are required to train this model. The Python 3 programming language was selected to launch the system. Its correctness was tested on the Windows 8.1, MacOS Catalina 10.15.6, and Ubuntu 20.04 LTS operating systems.

After passing the authorization procedure, the user is given unique identifier UID (UserID), which characterizes them in the system and maps their information in the database (data dynamics of the computer mouse handling and model-classifier). The SQLite library built into the programming language is used as a tool for working with the database. After passing the authorization procedure, the main loop of data collection, processing, storage, and dynamic decision-making on the legitimacy of the current user is started. Data are collected for training the model in 1 h of active work of the user with a computer mouse. The trained model is stored for future use. With a trained model in place, authentication is performed every 3 s, and the trust level of the model changes on the basis of the predictions of the SVC algorithm. If an intrusion is detected, a notification is sent to the control panel of the device with a warning. After 5 s, the entire system is blocked, after which reauthorization in the system and application is required.

A test mode was also used to simulate user operation by feeding events from test datasets and measuring performance by time windows:

- mean FRR = 0.089, median FRR ± IQR = 0.086 ± 0.002;
- mean FAR = 0.071, median FAR ± IQR = 0.070 ± 0.001;
- mean ANIA = 102 s, median ANIA ± IQR = 111 ± 36 s;
- mean ANGA = 118 min, median ANGA ± IQR = 117 ± 5 min.

The optimum threshold for calculating FRR (errors of the first kind) and FAR (errors of the second kind) is selected to minimize EER using the Python language library scikit-learn. Errors of the first kind are the proportion of cases in which the biometric system does not provide access to a legitimate user. Errors of the second kind are the proportion of cases in which the system grants access to an unauthorized person. EER is the equal rate of errors, which allows us to compare systems. A low EER means the system can be configured so that the rate of errors of both the first and second kind is lowest. We also measured the Average Number of Imposter Actions (ANIA, which indicates how many actions an attacker can perform before being blocked by the system and the Average Number of Genuine Actions (ANGA, which indicates how many actions an authenticated user can perform before being incorrectly blocked by the system). The values of ANIA and ANGA show that the developed system can be used in practice.

7. CONCLUSIONS

Dynamic user authentication based on an analysis of computer mouse handling is a promising field in developing modern means of authentication. It allows us to detect internal attacks in information systems in the shortest possible time, and at minimal expenditures on additional equipment. In this work, we proposed approaches to the preliminary processing of data and constructing feature space that include filtering collected data, gradient boosting for selecting the most important features, dynamic removal of local outliers, and normalization of the resulting feature space. A combination of these approaches helped improve the quality of recognition. Our experimental studies showed our version of dynamic user authentication based on such a fully convolutional neural network as an autoencoder produced results comparable in terms of accuracy to the using the SVC algorithm. The decision to block a current user is made according to a rapidly changing user trust level according to the Trust Model algorithm. A cross-platform dynamic user authentication system was designed and tested on the basis of our algorithm. Its performance indicators show it can be used for building promising modern information security systems that include tools for analyzing the dynamics of a user's computer mouse handling. We plan to investigate the stability of our approaches for different equipment in the future.

REFERENCES

1. A. K. Jain, A. Ross, and S. Prabhakar, "An introduction to biometric recognition," IEEE Trans. Circuits Syst. Video Technol. **14** (1), 4–20 (2004).
2. J. Wayman, A. Jain, D. Maltoni, and D. Maio, "An introduction to biometric authentication systems," in *Biometric Systems*, Ed. by J. Wayman et al. (Springer, London, 2005), pp. 1–20.
3. Y. X. M. Tan, A. Binder, and A. Roy, "Insights from curve fitting models in mouse dynamics authentication systems," in *Proc. 2017 IEEE Conference on Application, Information and Network Security* (*AINS*) (Miri, Malaysia, 2017), pp. 42–47.
4. A. A. Khalifa, M. A. Hassan et al., "Comparison between mixed binary classification and voting technique for active user authentication using mouse dynamics," in *Proc. 2015 International Conference on Computing, Control, Networking, Electronics and Embedded Systems Engineering* (*ICCNEEE*) (Khartoum, Sudan, 2015), pp. 281–286.
5. P. Chong, Y. Elovici, and A. Binder, "User authentication based on mouse dynamics using deep neural networks: A comprehensive study," IEEE Trans. Inf. Forensics Secur. **15**, 1086–1101 (2019).
6. P. Chong, Y. X. M. Tan et al., "Mouse authentication without the temporal aspect – What does a 2D-CNN learn?" in *Proc. 2018 IEEE Security and Privacy Workshops* (*SPW 2018*) (San Francisco, CA, 2018), pp. 15–21.

7. M. Antal and N. Fejér, "Mouse dynamics based user recognition using deep learning," Acta Univ. Sapientiae, Inf. **12** (1), 39–50 (2020).
8. S. Mondal and P. Bours, "A study on continuous authentication using a combination of keystroke and mouse biometrics," Neurocomputing **230**, 1–22 (2017).
9. C. Feher, Y. Elovici et al., "User identity verification via mouse dynamics," Inf. Sci.: Int. J. **201**, 19–36 (2012).
10. J. H. Friedman, "Greedy function approximation: A gradient boosting machine," Ann. Stat. **29** (5), 1189–1232 (2001).
11. M. Kazachuk, A. Kovalchuk, I. Mashechkin, I. Orpanen, M. Petrovskiy, I. Popov, and R. Zakliakov, "One-class models for continuous authentication based on keystroke dynamics," in *Intelligent Data Engineering and Automated Learning – IDEAL 2016*, *Proc. 17th Int. Conference*, Ed. by H. Yin et al., Lecture Notes in Computer Science, Vol. 9937 (Springer, Cham, 2016), pp. 416–425.
12. M. M. Breunig, H.-P. Kriegel et al., "LOF: identifying density-based local outliers," ACM SIGMOD Record **29** (2), 93–104 (2000).
13. F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *Proc. Eighth IEEE International Conference on Data Mining* (*ICDM 2008*) (Pisa, Italy, 2008), pp. 413–422.
14. B. Hoyle, M. M. Rau et al., "Anomaly detection for machine learning redshifts applied to SDSS galaxies," Mon. Not. R. Astron. Soc. **452** (4), 4183–4194 (2015).
15. M. Antal and L. Denes-Fazakas, "User verification based on mouse dynamics: A comparison of public data sets," in *Proc. 2019 IEEE 13th International Symposium on Applied Computational Intelligence and Informatics* (*SACI*) (Timisoara, Romania, 2019), pp. 143–148.
16. C. Shen, Z. Cai, and X. Guan, "Continuous authentication for mouse dynamics: A pattern-growth approach," in *Proc. IEEE/IFIP International Conference on Dependable Systems and Networks* (*DSN 2012*) (Boston, MA, 2012), pp. 1–12.

Translated by O. Pismenov